

软件设计与开发

ISSAC

施施

蔡俊哲

王渝

15307130017 15307130076 15307130258

目录

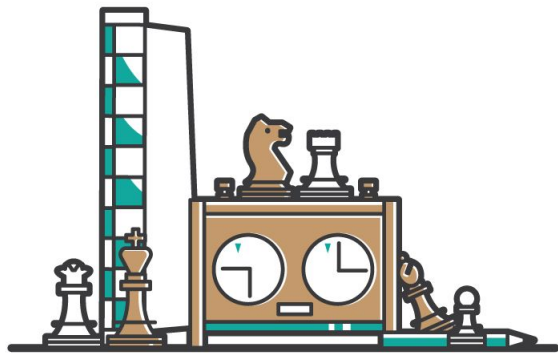
1 系统概述

2 系统需求说明

3 系统设计

4 分工情况与 git

5 DEMO





1 系统概述



系统概述

COCOS2DX

C++

Window macOS 跨平台

游戏说明

Issac 是 S.C.W 于 2018 年 06 月发行的一款 2D 平面角色扮演、动作冒险 RPG 类的独立游戏，主角 Issac 将在有着能够提升能力的道具与特殊技能的半 RPG 世界中闯荡。

游戏胜利条件：战胜迷宫阶层主蝇王。

游戏操作：WASD 控制 Issac 的移动，上下左右控制 Issac 的射击，E 键放下炸弹。在开始界面按 Enter 进入游戏模式，开始界面按 D 进入 Debug 模式。



2 系统需求说明



系统需求说明

开发工具

Windows: Visual Studio 2017 + Cocos2dx 3.16

macOS: Xcode 9.2 + Cocos2dx 3.16 + Sketch 48.2

运行环境

Windows 10 1803

macOS Sierra 10.12.6

依赖

glfw 3.1.2

关键技术

MVC 框架

单例模式

AOP 编程模型

BFS 最短路径算法

Prim 算法随机迷宫

数据抽象与面向对象技术

类接口技术

Lambda 表达式作为回调函数

工厂模式

装饰模式

物理引擎

节点树渲染



3 系统设计



系统设计

组织框架

Model

Character

运行逻辑与细节

ViewModel

物理引擎

Listener

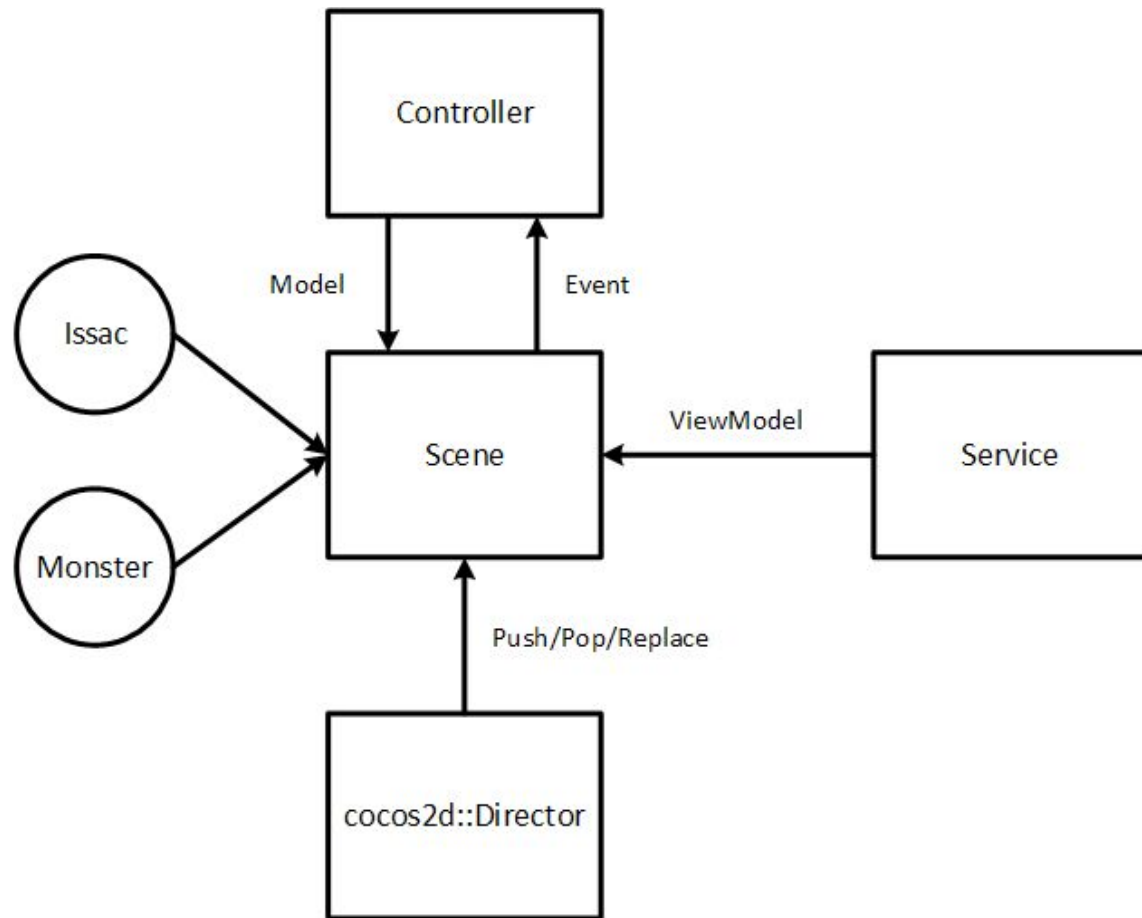
Service

Resource 组织结构

Controller

Scene

组织框架



运行逻辑与细节

- 软件运行后进入欢迎界面, 用户输入 Enter 后, 进入主菜单。
- 在主菜单用户可以通过上下方向键控制选择光标, 输入 Enter 后游戏开始, 主菜单销毁。
- 游戏主界面为空房间, 房间地面背景上有基本操作提示, 用户输入 WASD 分别控制 Issac 上下左右运动, 输入方向键 $\uparrow\downarrow\leftarrow\rightarrow$ 分别控制 Issac 发射 Tear, 输入 E 控制 Issac 放置炸弹。
- 所有的房间均有门通向其他房间, Issac 进入其他房间后, 当前房间销毁, 新房间生成, 但 service 保存所有房间信息, Issac 返回后, 原始房间信息还原。

运行逻辑与细节

- Issac 自身游戏属性有:生命值、移动速度、攻击力、Tear 射速、Tear 射程、炸弹数、子弹击中石头和墙壁后是否可以回弹、是否可以飞行。
- 房间顶层有 HUD 面板, 显示用户生命值、可放置的炸弹数, 如用户没有炸弹仍然放置, 则炸弹数闪烁以通知用户。有半透明小地图(透明度可调), 显示用户当前房间和相邻房间(可区分普通房间、宝藏房间和 boss 房间), Issac 当前所处房间始终居中。

运行逻辑与细节

- 在任一游戏运行界面若用户输入 Esc 键, 游戏界面暂停(音乐仍然播放)并显示暂停菜单, 暂停菜单上显示 Issac 游戏属性信息(实时更新)。用户可以输入方向键 $\uparrow\downarrow$ 控制暂停界面光标, 输入 Enter 跳转到响应界面。若用户选择 resume, 则游戏继续进行。若用户选择 exit game, 则游戏退出, 当前游戏销毁并显示主菜单。若用户选择 options, 则显示设置界面。

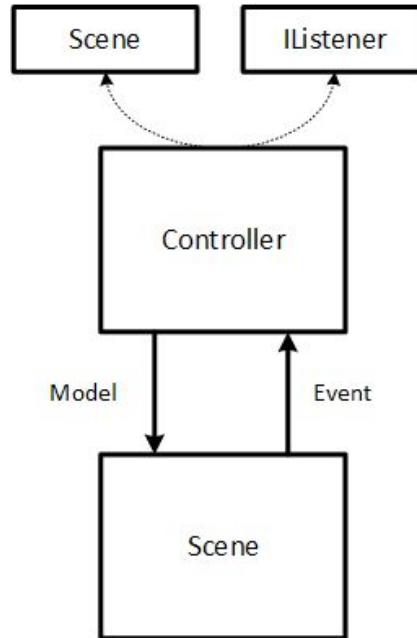
运行逻辑与细节

- issac 可拾取地面物品，拾取物品后 issac 游戏属性变化。issac 与怪物或怪物的 Tear 碰撞，或在炸弹爆炸范围内时，受到伤害，生命值减少（不同伤害生命值减少不同），issac 角色闪烁以提示用户，并播放受伤音效。怪物在死亡后显示死亡动画和地面特效。issac 可飞行后，可以在石头上运动。
- 炸弹放置后闪烁一定时间，然后显示爆炸效果，在一定爆炸范围内的怪物或 issac 受到伤害。爆炸后地面显示效果。

运行逻辑与细节

- 在设置界面, 用户输入方向键 $\uparrow\downarrow$ 控制设置界面光标, 输入方向键 $\leftarrow\rightarrow$ 调节对应设置项数值。可调设置项有: 音量、音效、小地图透明度。
- 主菜单、游戏界面、游戏失败界面背景音乐各不相同。

Listener - Controller - Model



Listener

Listener 下共有两个文件, IMainSceneListener.h 和 IRoomSceneListener.h。IListener 类中声明 Controller 所用到的函数接口。

1. on_key_pressed
2. on_key_released

Controller

Controller 负责初始化场景, 初始化音乐引擎, 设置事件响应, 处理用户的输入并更新 model。Controller 与 View 没有直接数据交换。Controller 继承 Scene 类和 IListener 接口。

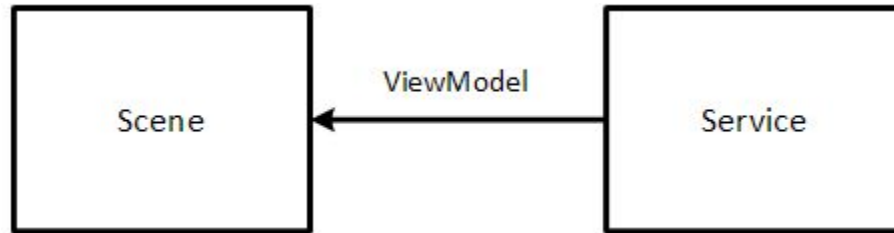
- 1. MainSceneController**
- 2. RoomSceneController**

Model

Model是Controller向Scene传递信息的载体。

- 1. MainSceneModel中包含与主界面菜单相关的数据。**
- 2. RoomSceneModel中包含玩家移动和攻击的数据和一些内嵌菜单状态的相关数据。**

ViewModel - Service



ViewModel

ViewModel由Service创建, 传递给Scene, 然后Scene根据ViewModel条件渲染。

- 1. RoomViewModel**
- 2. PlayerViewModel**
- 3. MiniMapViewModel**

ViewModel - RoomViewModel

采用“工厂”设计模式

```
static RoomViewModel createRoomViewModel(int roomType, bool visited, int barrierType, bool item_taken);
```

输入：房间编号、是否被访问过、地形编号和是否拿取物品

输出：房间具体布置

房间编号	房间内容
roomType:0	初始房间
roomType:1~7	怪物房间。1:全Fatty, 2:全Fly, 3:全Gaper, 4:全Spider, 5:Spider+FattyFire, 6:Fly+FlyFire, 7:Fly+GaperFire
roomType:8~20	宝藏房间。宝藏房号20是测试用, 会将所有宝藏放在房间里
roomType:21+	代表Boss房,21:Boss-flyDaddy
地形编号	地形内容
barrierType:0	空地
barrierType:1	在中央小石头排成十字形
barrierType:2	四块大石头

ViewModel - RoomViewModel

房间具体布置:

将房间可移动部分划分成一张13*7的棋盘

内容编号:0表示空地, 1表示小石头, 2表示大石头, 3表示玩家, 4~19表示怪物的各种类别, 20~30表示宝藏

ViewModel - MiniMapViewModel

小地图的信息

小地图是5×5的简单棋盘，Service在遍历自身维护的地图表后，会生成5×5二维数组，二维数组有房间对应位置上的值是roomType和地图定义保持一致。

ViewModel - PlayerViewModel

玩家属性

- 生命值
- 攻击值
- 移动速度
- 眼泪速度
- 眼泪射程
- 眼泪间隔时间
- 碰撞半径
- 质量
- 飞行和弹跳
- 炸弹数量

Service

Service采用“单例”设计模式

```
static RoomService *getInstance();
```

有RoomService和PlayerService两个主要类

Service

RoomService负责管理房间, **PlayerService**负责管理玩家的状态。首先, 游戏启动时, **RoomService**内部维护一个map, 生成地图, 这个地图可以是随机生成的迷宫, 也可以是手动设计的地图。map里的数据结构是地图服务的核心, 所有辅助函数都会访问这个结构的成员。

RoomService

类型	成员	注释
int	current_room_id	当前房间ID
int	left_room_id	左边房间ID, 如果不存在则为0
int	up_room_id	上边房间ID, 如果不存在则为0
int	right_room_id	右边房间ID, 如果不存在则为0
int	down_room_id	下边房间ID, 如果不存在则为0
bool	visited	房间是否被访问过
bool	item_taken	房间里的物品是否被拿走
int	current_room_type	房间类型, 和上文一致
int	current_barrier_type	石头的生成形状, 和上文一致

RoomService

Debug模式固定地图

游戏界面按"D"进入Debug模式



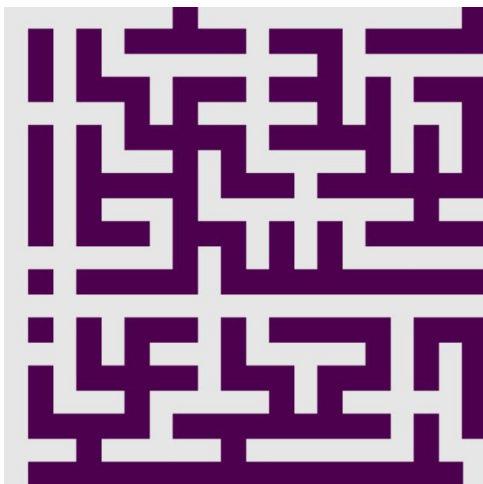
RoomService - 正式模式的复杂迷宫

正式模式地图: 10*10的迷宫

初始: 迷宫左下角, 且出生地边上一定有一个宝藏房间

Boss: 迷宫右上角, 打败boss即可通关游戏。

迷宫地形通过prim算法随机生成。



RoomService - 正式模式的复杂迷宫

在迷宫生成后, 通过循环遍历每个房间来将迷宫转换成上述定义的map类型。

房间四周的RoomID可通过访问迷宫得到。

房间初始是否访问=false。

房间初始是否拿取物品=false。

房间的地形和类型完全随机, 三种地形等概率出现, 房间类型有5%的几率为宝藏房间, 有95%几率为怪物房间。

RoomService - 小地图生成、音乐

小地图的生成算法是两层遍历，先遍历当前房间的所有方向，然后再遍历四个方向房间各自的四个方向房间，如果，房间没有访问到是不会进行二次遍历（一次遍历始终进行），最终会生成5×5的二维数组数据。在Scene中会根据这个二维数组加载正确的贴图到正确的地方。

- RoomService还管理了背景音乐的大小和小地图的透明度。
- RoomService提供了许多辅助方法，方便其他类调用Service。

PlayerService - 玩家管理

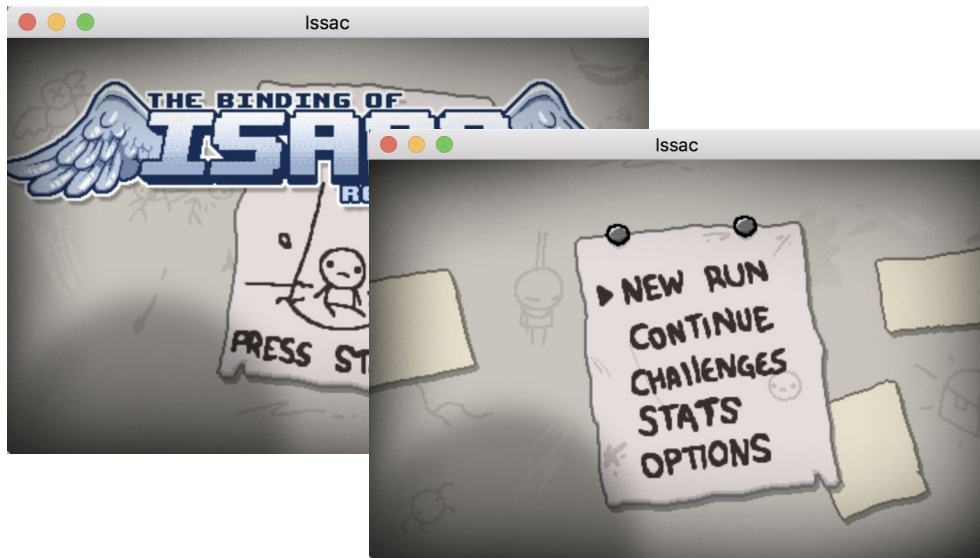
PlayerService比RoomService简单，主要负责保存玩家属性，另外也提供了许多辅助函数方便调用。

Scene

Scene 负责绘制场景和场景内所有物体的操作。

- 1. MainScene**
- 2. RoomScene**

Scene - MainScene



Scene - RoomScene

大量成员

Update 逻辑

房间生成

物体控制

场景绘制

类型	名称	说明	Zorder 或容器内项目 Zorder
RoomSceneModel	model	RoomScene 的 model	无
Issac *	player	玩家控制的角色	3
cocos2d::Sprite *	room_piece1	房间地面左上角	0
cocos2d::Sprite *	room_piece2	房间地面右上角	0
cocos2d::Sprite *	room_piece3	房间地面左下角	0
cocos2d::Sprite *	room_piece4	房间地面右下角	0
cocos2d::Sprite *	controls	初始房间地面上的操作提示	0
cocos2d::Sprite *	shading	房间光线阴影	2
cocos2d::Sprite *	overlay	房间全局光线遮罩	6
cocos2d::Sprite *	edgeSpace	房间的周围边界(物理引擎检测用)	默认值
cocos2d::Sprite *	tearSprite	issac 发射的 Tear	默认值
cocos2d::Sprite *	pausescreen	暂停菜单	7
cocos2d::Sprite *	optionscreen	设置菜单	7
cocos2d::Sprite *	deadscreen	死亡菜单	9
cocos2d::Sprite *	winscreen	获胜菜单	10

cocos2d::SpriteFrame *	fullheart	healthbar 的元素	无
cocos2d::SpriteFrame *	halfheart	healthbar 的元素	无
cocos2d::Sprite *	bosshealthbar	boss 血条	6
cocos2d::Sprite *	minimap	小地图	5
cocos2d::Sprite *	healthbar	玩家血量 HUD	8
cocos2d::Sprite *	hud_bomb	玩家炸弹 HUD 图标	8
cocos2d::Label *	Count_bomb	玩家炸弹数标签	8
cocos2d::Sprite *	bomb	炸弹	3
cocos2d::Sprite *	bombraduis	炸弹弹坑	2
cocos2d::Sprite *	blood_pool	血迹	2
cocos2d::Sprite *	ghost_sprite	issac 死亡后的幽灵	3
cocos2d::Sprite *	deadbody	issac 死亡的尸体	3
cocos2d::Sprite *	collectible_streak	拾取物品提示	6
cocos2d::Sprite *	temp_sprite	引导 Tear 播放销毁动画的1x1透明贴图	3

cocos2d::Vector<Monster*>	monsters_	容器, 存放怪物指针	3
cocos2d::Vector<Tear*>	tears_	容器, 存放 Tear 指针	默认值
cocos2d::Vector<Stone*>	stones_	容器, 存放石头指针	2
cocos2d::Vector<Door*>	doors_	容器, 存放门指针	1
cocos2d::Vector<Collectable*>	collectables_	容器, 存放拾取物品指针	3
int	prev_bomb_num	前一帧的炸弹数	无
IRoomSceneListener *	listener_	设置事件监听	无
MiniMapViewModel	mini_map_vm_	小地图 Viewmodel	无
RoomViewModel	room_vm_	房间 Viewmodel	无

Scene - RoomScene

房间生成:

roomService中的RoomViewModel

RoomViewModel的13*7棋, 放上对应的物体(怪物、玩家、石头、门)生成无形边界。

Scene - RoomScene

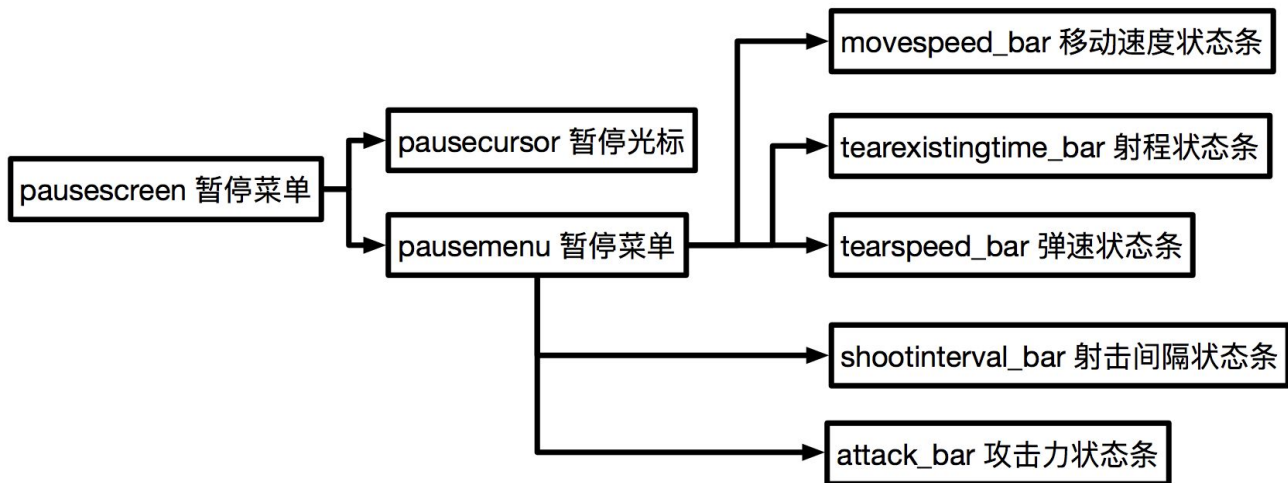
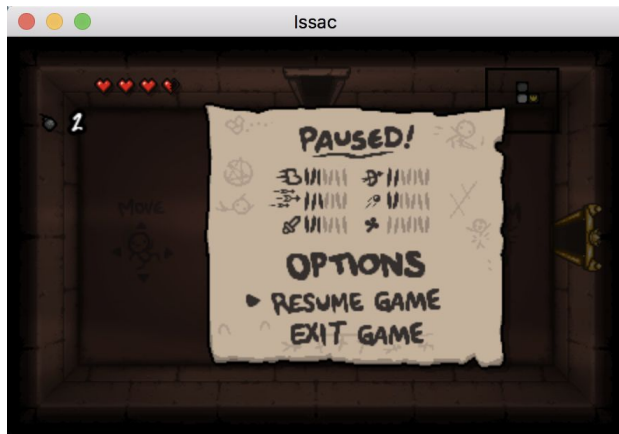
物体控制: update()中按帧进行计算

子弹: 存在时间

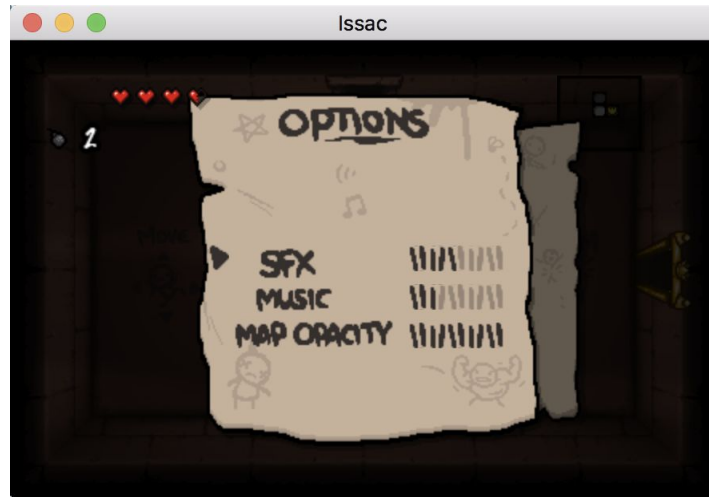
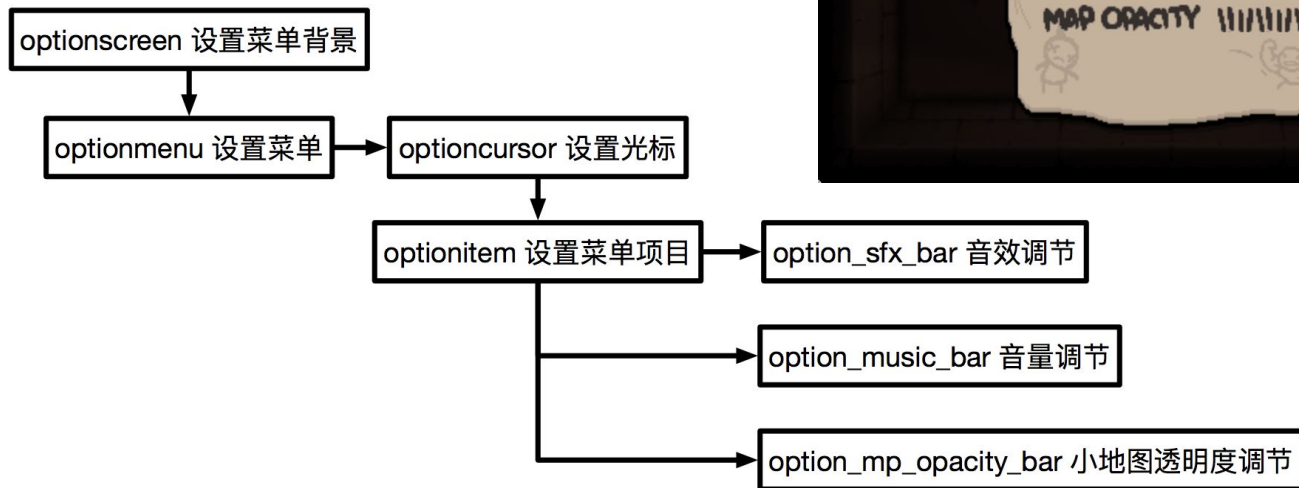
怪物: 血量、移动、射击、生怪物

玩家: 血量、移动、射击、放炸弹

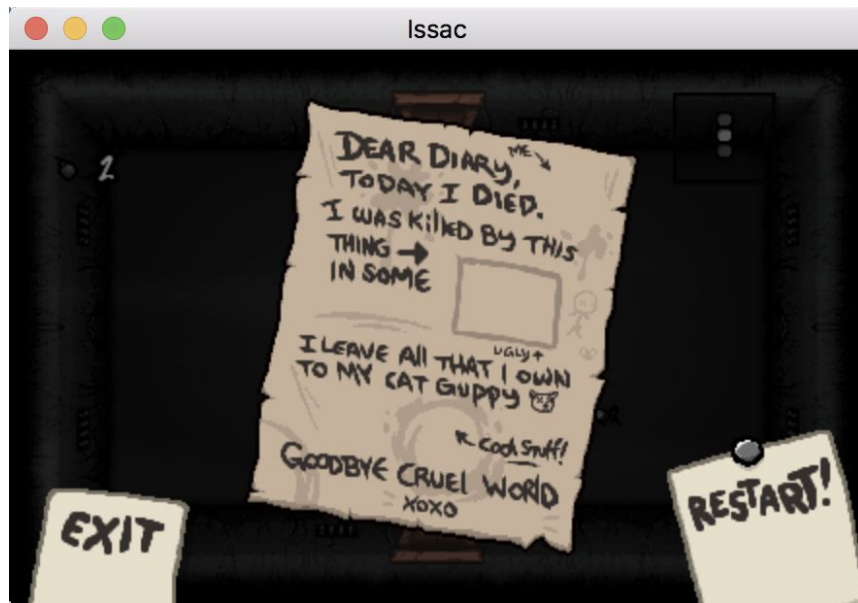
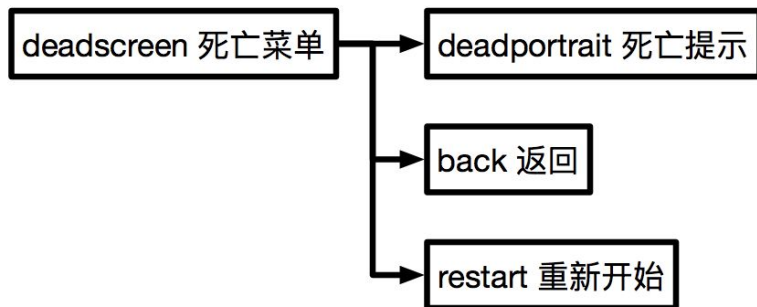
Scene - RoomScene



Scene - RoomScene



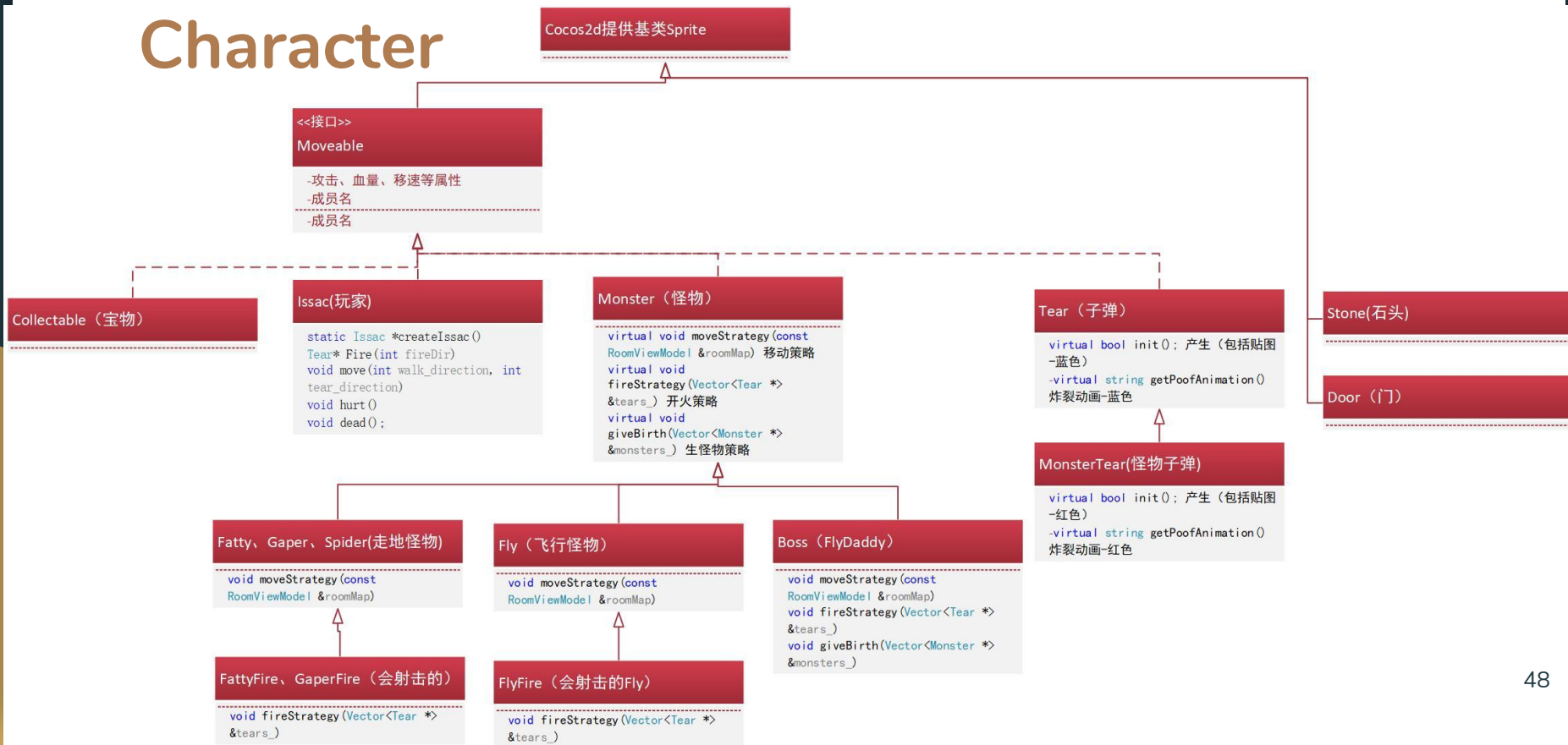
Scene - RoomScene



Scene - RoomScene



Character



基类/接口类Moveable

可动物体的基类/碰撞检测的接口类

记录属性：

血量，攻击，移速，子弹速度，子弹射程，身体大小

.....

玩家类Issac

移动

射击

受伤动画

死亡动画


怪物策略

移动策略: 随机移动, 简单移动, 冲刺+停顿移动, 最短路径移动(BFS), 最短路径移动+变速

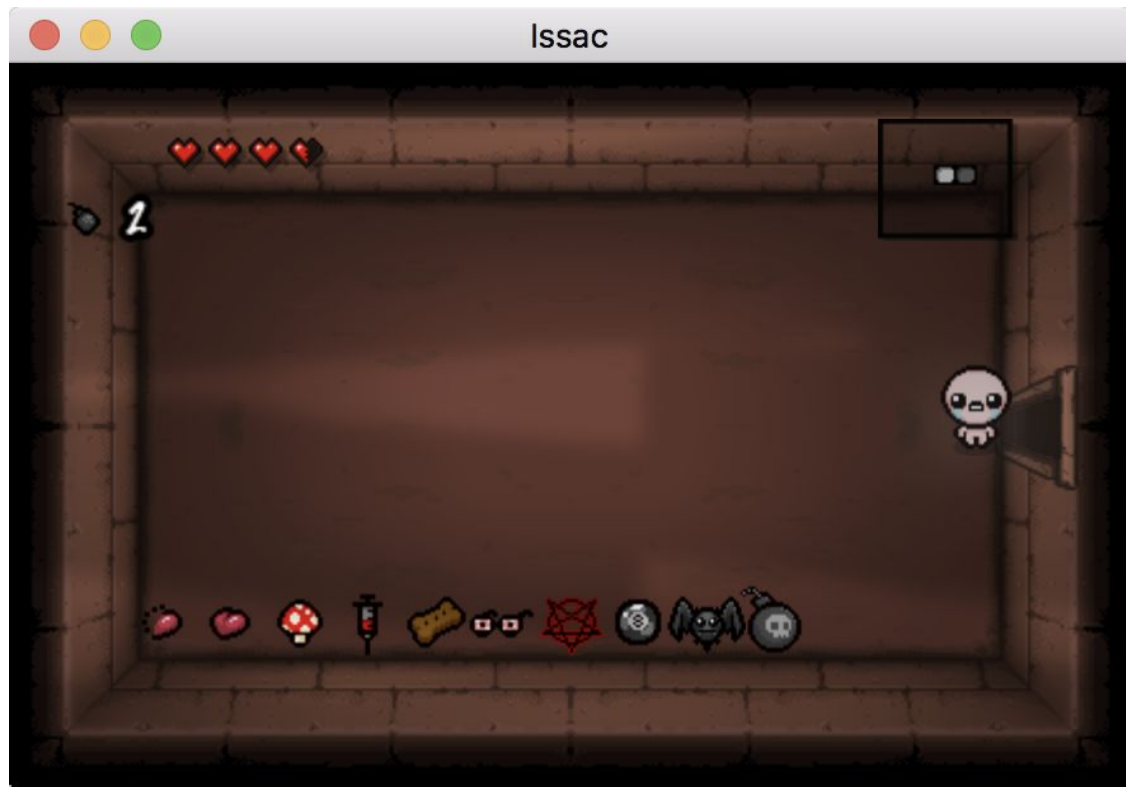
开火策略: 指向射击, 十字射击, X字射击

生怪物策略(Boss): 产生苍蝇, 产生射击苍蝇

怪物名	移动策略	开火策略	图片
Fatty	最短路径移动	-	
FattyFire	随机移动	十字射击	
Fly	简单移动(飞行怪物不会碰撞石头)	-	
FlyFire	简单移动(飞行怪物不会碰撞石头)	指向射击	
Gaper	最短路径移动+变速	-	
GaperFire	最短路径移动+变速	X字射击	
Spider	冲刺+停顿移动	-	

Boss	策略	描述
FlyDaddy 	移动策略	1/5几率向玩家冲刺, 其余以 缓慢速度在房 间中移动
	射击策略	如果没有冲刺, 1/4几率向四周 8个方向发射3轮间隔短的子弹
	生怪物策略	如果没有冲刺, 也没有射 击, 1/2几率产生3个苍蝇Fly, 1/2几率产生1个射击苍蝇FlyFire

可拾取物品



宝物名	宝物编号	宝物属性	宝物图片
半颗心	0	加半颗血 (health+1)	
一颗心	1	加一颗血 (health+2)	
蘑菇	2	加攻击0.3, 加血半颗心, 略微加移动速度, 射速, 射程	
不明药剂	3	加攻击3, 扣1颗心的血, 减慢移动速度	
骨头	4	加攻击1	
眼镜	5	减攻击0.7, 射程几乎无限	
恶魔召唤	6	加攻击4, 射程减半	
弹球	7	子弹现在反弹了	
恶魔之翼	8	人能够飞了 (不和石头碰撞)	
炸弹	9	增加一个炸弹	

物理引擎

创建: Scene的物理世界 & Moveable的物理身体

移动: 步进

Issac和Monster类的速度控制

物理引擎

碰撞: 类别掩码categoryBitmask和检测掩码contactTestBitmask

$(A \rightarrow \text{categoryBitmask} \ \&\& \ B \rightarrow \text{collisionBitmask}) \ || \ (B \rightarrow \text{categoryBitmask} \ \&\& \ A \rightarrow \text{collisionBitmask})$

碰撞检测: 类别掩码categoryBitmask和碰撞掩码collisionBitmask

$(A \rightarrow \text{categoryBitmask} \ \&\& \ B \rightarrow \text{contactTestBitmask}) \ || \ (B \rightarrow \text{categoryBitmask} \ \&\& \ A \rightarrow \text{contactTestBitmask})$

种类	CategoryBitmask	碰撞规则	CollisionBitmask
玩家	0000_0000_0001(001)	所有	1000_1111_1111(8FF)
走地怪物	0000_0000_0010(002)	所有	1001_1111_1111(9FF)
飞行怪物	0000_0000_0100(004)	不和普通石头碰撞	1001_1101_1111(9DF)
子弹	0000_0000_1000(008)	所有	1111_1111_1111(FFF)
边界	0000_0001_0000(010)	不和门、无形边界石头、普通石头碰撞	1111_1000_1111(F8F)
石头	0000_0010_0000(020)	不和门、无形边界石头、普通石头、飞行怪物和飞行玩家碰撞	1010_1000_1011(A8B)
门	0000_0101_0000(050)	不和门、无形边界石头、普通石头碰撞	1111_1000_1111(F8F)
宝藏	0000_1000_0000(080)	所有	1001_1111_1111(9FF)
飞\玩家	0001_0000_0000(100)	不和普通石头碰撞	1000_1101_1111(8DF)
无敌\玩家	0010_0000_0000(200)	不和走地，飞行怪物碰撞	1000_1111_1001(8F9)
无敌飞\玩家	0100_0000_0000(400)	不和走地，飞行怪物，普通石头碰撞	1000_1101_1001(8D9)
反弹子弹	1000_0000_0000(800)	所有	1111_1111_1111(FFF)

检测掩码	低位											高位
种类	玩家	走地怪物	飞行怪物	子弹	边界	石头	门	宝藏	飞/玩家	无敌/玩家	无敌飞/玩家	反弹子弹
玩家		1	1	1			1	1				1
走地怪物	1			1					1			1
飞行怪物	1			1					1			1
子弹	1	1	1	1	1	1	1	1	1	1	1	1
边界				1								
石头				1								
门	1			1					1	1	1	
宝藏	1			1					1	1	1	
飞\玩家		1	1	1			1	1				1
无敌\玩家				1			1	1				1
无敌飞\玩家				1			1	1				1
反弹子弹	1	1	1	1					1	1	1	1

物理引擎

碰撞处理: 监听器

`onContactBegin(const PhysicsContact& contact)`

(a) 通过两个碰撞对象的Tag判断身份

(b) 根据身份进行操作

Resource 组织结构

目录	子目录	功能
font		存放软件 GUI 用到的全部字体
gfx		存放软件 GUI 用到的全部贴图, 该级目录下有部分常用贴图
	backdrop	存放游戏各房间的背景贴图、房间全局阴影遮罩贴图、主角高光贴图
	bosses	存放游戏 boss 贴图
	characters	存放主角 issac 贴图
	effects	存放图形特效贴图, 如爆炸效果、血迹
	grid	存放房间的门、石头的贴图
	items	存放可拾取物品的 贴图
	monsters	存放普通怪物的 贴图
	overlays	存放每种房间的光线阴影遮罩 贴图
	ui	存放所有菜单贴图
music		存放软件用到的全部背景音 乐
sfx		存放软件用到的全部音 乐特效



4 分工情况与 git



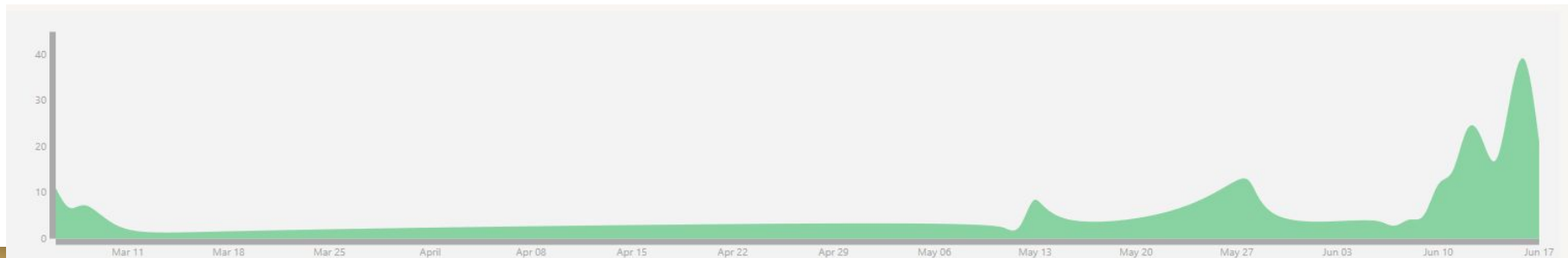
分工情况与 git

273 Commits
2W+

施施 前期调研, 键盘控制逻辑, 界面贴图和动画, 房间渲染, 菜单逻辑, 游戏音效

蔡俊哲 房间物体生成, 物体控制, 游戏逻辑, 物理引擎集成, 随机地图生成

王渝 服务逻辑, MVC框架, 房间系统, 小地图, git管理





5 DEMO

